Geo

```
###----------------------------------------------------------------------------------
###----------------------------------------------------------------------------------
### Geostatistics
###----------------------------------------------------------------------------------
###----------------------------------------------------------------------------------

library(sp)
library(lattice)
library(geoR)
library(gstat)
v.f     <- function(x, ...){100-cov.spatial(x, ...)}
v.f.est <- function(x,C0, ...){C0-cov.spatial(x, ...)}

# ---------------------------------------------------------------------------------
# EXPLORATORY ANALYSIS
# ---------------------------------------------------------------------------------
data(meuse)
data = meuse
data = data[,c(1,2,6,14)]

# rinominiamo
colnames(data) = c('x','y','Z','dist')

# coordinates
coordinates(data) = c('x', 'y')
head(data)

# bubble plot
bubble(data, 'Z', do.log=TRUE, key.space='bottom')

# histogram of Z
hist(data$Z, breaks=16, col="grey", main='Histogram of Z', prob=T, xlab = 'Z')

# Comment
# If it's highly skewwd, transform to the log
hist(log(data$Z), breaks=16, col="grey", main='Histogram of log(Z)', prob=T, xlab='log(Z)')

## From here we go on with Z, not log(Z)
# check if distance is influenced
xyplot( Z ~ sqrt(dist), as.data.frame(data), col='black', pch=19)

# Comment
# is there positive/negative correlation?

# ---------------------------------------------------------------------------------
# ESTIMATING SPATIAL CORRELATION VARIOGRAM ANALYSIS
# ---------------------------------------------------------------------------------
# sample variogram (binned estimator)
# (NB: it ignores the directions)
svgm = variogram(Z ~ 1, data)
plot(svgm, main = 'Sample Variogram',pch=19)

# Comment
# Does it look like a stationary variogram?

# let's add directions
plot(variogram(Z ~ 1, data, alpha = c(0, 45, 90, 135)),pch=19)

# Comment
# Does it change? Does it seem like anisotropy?
# Maybe the variogram is different in the asimptote

# Let's go on even if there is anisotropy
#   plot(variogram(Z ~ 1, data, cutoff = 1000, width = 1000/15),pch=19)
# intervals can have different widths: to fix varying widths use the argument boudaries
#   plot(variogram(Z ~ 1, data, boundaries = c(0,200,seq(400,1500,100))),pch=19)

# ---------------------------------------------------------------------------------
# VARIOGRAM MODELING
# ---------------------------------------------------------------------------------
# list of parametric isotropic variogram models
vgm()
# vgm(still, model, range, nugget)

# spherical model
vgm(1,'Sph',300)

# spherical model with a nugget
vgm(1,'Sph',300, 0.5)

## weighted least squares fitting a variogram model to the sample variogram
## STEPS:
## 1) choose a suitable model
## 2) choose suitable initial values for partial sill, range & nugget
## 3) fit the model using one of the possible fitting criteria
```

```
v  = variogram(Z ~ 1, data)
v2 = variogram(Z ~ sqrt(dist), data)
plot(v,pch=19)
plot(v2, pch=19)

# Linear behavior near the origin, growth not very fast
# Recall: both spherical and exponential model have a linear behavior near the
#         origin but exponential model has a faster growth than the spherical one
# => we fit a spherical model

# Try reasonable initial values
# Comment
# Se esce:
# Warning in fit.variogram(v, vgm(1, "Sph", 10, 1)): singular model in variogram
# stai sbagliando variogram

# plot of the final fit
v.fit  <- fit.variogram(v,  vgm(0.6, "Sph", 0.5, 0.1))
v.fit2 <- fit.variogram(v2, vgm(1, "Sph", 0.5))
plot(v, v.fit, pch = 19)
plot(v2, v.fit2, pch=19)

# fitting method: non linear regression with minimization of weighted
# sum of squares error. final value of the minimum
attr(v.fit, 'SSErr')
attr(v.fit2, 'SSErr')

# ------------------------------------------------------------------------------
# SPATIAL PREDICTION & KRIGING
# Let's assume that our field is isotropic and stationary
# ------------------------------------------------------------------------------
# Stationary Univariate Spatial Prediction (Ordinary Kriging)
# ------------------------------------------------------------------------------
# Prediction in a single new location
s0.new = data.frame(x=77.69, y=34.99)
coordinates(s0.new)=c('x','y')

plot(data, pch=19)
plot(s0.new, col='red', add=T, pch=16)

# Create a gstat object setting a spherical (residual) variogram
g.tr <- gstat(formula = Z ~ 1, data = data, model = v.fit)

## ORDINARY KRIGING
# Ordinary kriging prediction with: predict(obj, grid, BLUE=FALSE)
# (gives the prediction of Z(s_0))
predict(g.tr, s0.new)                                                # Z*
# Comment
#  * var1.pred = prediction (Z*)   !!! REMEMBER IF IT'S THE LOG
#  * var2.pred = variance of the prediction

# Estimate the mean: use the argument 'BLUE'
predict(g.tr, s0.new, BLUE = TRUE)                                   # E[Z]
# Comment
#  * var1.pred = estimate of the mean
#  * var2.pred = variance of the estimation of the mean
# this gives the estimate of the mean under gls

# Comment
# the prediction of Z(s_0) of a point where we observe data gives zero variance,
# but the prediction of the mean has the variance

# ------------------------------------------------------------------------------
# Non-stationary Univariate Spatial Prediction (Universal Kriging)
# Let's see if the distance has a meaning
# ------------------------------------------------------------------------------
# Create a gstat object setting a spherical (residual) variogram
g.tr2 <- gstat(formula = Z~sqrt(dist), data=data, model=v.fit2)
g.tr2

v.gls     <- variogram(g.tr2)
v.gls.fit <- fit.variogram(v.gls, vgm(1, "Sph", 0.5))
plot(v.gls, v.gls.fit, pch = 19)

# Update gstat object with variogram model
g.tr2 <- gstat(formula = Z ~ sqrt(dist), data = data, model=v.gls.fit)

## UNIVERSAL KRIGING
# We have to define the covariate in s_0
s0.vec <- as.vector(slot(s0.new,'coords'))

# Distance
s0.new              <- data.frame(x=77.69, y=34.99)
coordinates(s0.new) <- c('x','y')
s0.dist             <- 1
s0.new              <- as.data.frame(c(s0.new,s0.dist))
names(s0.new)       <- c('x','y','dist')
```

```r
coordinates(s0.new) <- c('x','y')
s0.new              <- as(s0.new, 'SpatialPointsDataFrame')
s0.new

# Function "predict" uses the residual variogram stored in the gstat
# object to make the prediction
predict(g.tr2, s0.new)

# this gives the estimate of x(s_0)'*beta (trend component) under gls
# (estimate of the drift)
predict(g.tr2, s0.new, BLUE = TRUE)

# ------------------------------------------------------------------------------------
# Evaluating coefficients
# ------------------------------------------------------------------------------------
# Model:   Z(S_i) = b_0 + b_1 * dist(s_i) + eps(s_i)

# b_0: trovo un punto a distanza zero
zero = data[which(data$dist == 0),]
zero = zero[1,]
zero
b_0 = predict(g.tr2, zero, BLUE = TRUE)$var1.pred

# b_1: trovo un punto a distanza qualsiasi dopodiche divido per la distanza
uno  = data[which(data$dist != 0),]
uno  = uno[1,]
dist = uno$dist
b_1  = (predict(g.tr2, uno, BLUE=TRUE)$var1.pred - b_0)/dist

# alternativamente
z0.new              <- data.frame(x=s.0$x, y=s.0$y, dist=0)
z0.new[2,]          <- data.frame(x=s.0$x, y=s.0$y, dist=1)
coordinates(z0.new) <- c('x','y')
z0.new              <- as(z0.new, 'SpatialPointsDataFrame')
z0.new
beta = predict(g.tr2, z0.new, BLUE=TRUE)$var1.pred
beta_0 = beta[1]
beta_1 = beta[2]-beta_0
```